Computer Science 773

Operating Systems

1998 EXAMINATION : ANSWERS AND NOTES.

As usual, not *the only* correct answers. Comments added after the event are in this typeface.

PART A.

QUESTION 1.

In an open-loop control system, the control output is calculated from the required set-point and any relevant environmental inputs, but without feedback from the controlled system itself. Typically, a calibration function evaluated from previous experiments with the system is used to calculate the control output.

Everyone coped pretty well with this bit.

The advantages of an open-loop control system are cheapness (no sensory equipment is required), speed (the control output is evaluated directly from the inputs without any time lag from the plant's slow response), and adequacy (in many cases, the control exercised by a closed-loop function is sufficiently good to be acceptable).

The disadvantages of an open-loop system are that it cannot react to any disturbances which have not been foreseen and (usually) measured beforehand.

I did want at least one advantage and one disadvantage.

An adaptive system incorporates a higher-level controller and an observer. The controller typically alters parameters in a lower-level controller to cope with changes in the overall performance reported by the observer. If the lower-level system were an open-loop controller, the result would be an adaptive open-loop controller.

Usually sensibly answered.

Such a system exhibits an element of paradox because feedback is required by the adaptive component, but ignored by the direct controller; the major advantage of an open-loop controller – the cost – is, apparently, not gained. The paradox is resolved if the form or timing of the feedback is not such as could be used in the direct controller. An example might be a machine designed to throw objects at a target; simple dynamics might give a satisfactory function for an open-loop controller to work quite well, but could not make allowance for wind speed; an adaptive controller could observe how closely the target was approached, and use the information to amend parameters in the open-loop system.

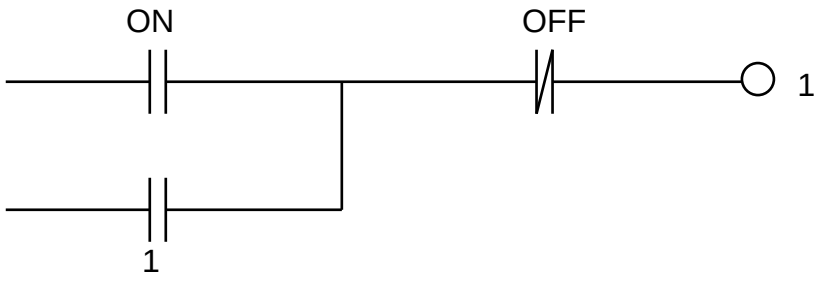
Not many answered the bit about "circumstances in which such a system could be effective even if a closed-loop controller could not be used".

QUESTION 2.

This wasn't a popular question. I'm surprised.

A ladder logic diagram is a graphical way of composing a logic programme. Each "rung" of the diagram identifies a relay circuit which implements a logical function of one or more variables, with AND operators implemented by serial connections, OR by parallel connections, and negation by "normally-closed" relays. All rungs operate continuously and in parallel. The input variables for a rung may be system inputs from the external world, or may be the values computed by rungs in the ladder. The outputs from the rungs (normally shown at the right-hand end) can be used as input variables for other rungs, and may also be system outputs.

I expected at least two or three of the points made in that specimen answer, and sometimes got them. Were people frightened off by my request for the "***nature*** of ladder-logic diagrams" ? I don't think you should jib at such things at graduate level.



Output : 1

Initially, the output is off and the buttons are in their normal states.
When ON is pressed, the circuit through ON and the normally-closed OFF is made, so output 1 (which is also the power output) is energised. The relay 1 now closes, so that the ON button is bypassed and can be released without affecting the power supply.
On pressing OFF, the circuit is broken, output 1 is turned off, relay 1 opens, and the ON bypass is therefore blocked. OFF can now be released, and the system returns to its initial state.
Because OFF completely controls the output, while the button is pressed it doesn't matter what you do with ON; the power is turned off.

This was sometimes not well done. I'm even more surprised; I thought it was really too simple, but maybe you'd like some easy marks. And even when it was well done, it wasn't always well described. I asked for the description just to make sure that the diagram wasn't an accident; perhaps I found out.

QUESTION 3.

This wasn't a popular question either, but I'm not so surprised. It was usually chosen last, commonly a sign of desperation.

	Machine tools	Manual operation
→	Servo-controlled tools	Manual operation
→	NC tools	Specialised digital controller, paper tape
→	DNC tools	Specialised controller, connection to remote computer
→	CNC tools	Computer integrated with machine tool

CNC tools are economical to use because, once programmed, they can repeat operations much faster than human operators can. This saves both by getting more work out of the machine, and by requiring fewer operating staff. Part programmers must be employed, but they write the programme only once, whereas the machine operators in the manual system were occupied in producing every part.

A part-programming language is used to encode all the details of a tool's activity in a manufacturing operation. It must describe the complete path of the machine's cutting tool in the manufacturing process. One would expect data types appropriate to the representation of the geometry of the system (points, lines, curves, angles, solid bodies) and to the representation of tool motion (tool choice, cutting speed, free motion speed, etc.).

QUESTION 4.

This was a popular question. I'm surprised. It was usually rather well answered, though some of the answers seemed to me to get symptoms under the wrong heading – such as people not bothering to check things in case (b), which I'd rather put in case (c). Clearly enough, though, some such decisions are matters of judgment, and I usually accepted them.

(a) The operator can spend too much time repeating checks which the plant carries out satisfactorily, or searching for information about the plant which he thinks he should know. In so doing he can miss real problems which should be dealt with.

Remedies include anything which will increase the operator's confidence in the plant. All plant variables should be accessible without difficulty at any time, and other technically sensible measures of good performance displayed.

(b) The operator can become bored, become absorbed in other tasks, or fall asleep.

Specific exercises for the operator can be built into the system : manual control periods, requirements for periodic checks and reports, emergency drills.

(c) The operator can become careless, and issue instructions to the plant which might be dangerous in the belief that the system will prevent any serious consequences.

Where possible, the system can monitor the operator's actions and initiate dialogue in cases where reasonably expected consequences appear to be dangerous. It is possible to make the system override the operator's actions, but you have to be very confident about your system before doing that.

QUESTION 5.

This was a deeply unpopular question, but (except in one case) very well done. I'm not sure whether I'm surprised or not. As computists, you could reasonably be expected to be interested in the process of designing and constructing software, but it isn't a thing we push a lot in the department. (I think that's a bad thing, but there we are.)

There are four interdependent components to the conversion; I describe them separately, but in practice they are linked in many ways, and are likely to proceed to some extent in parallel.

Step 1 : Producing the fabrication programme.

By considering the product specification, a sequence of operations which will construct it is defined. This is expressed in terms of physical operations on the parts of the product.

Step 2 : Design the manufacturing plant.

The equipment required to manufacture the product is designed. This will perhaps include several tools of various types, and means of moving the workpiece between the tools.

Step 3 : Define the manufacturing sequence.

Given the fabrication programme and the plant design, a sequence of machine instructions which will cause the plant to execute the fabrication programme is determined. This is now in terms of instructions which the machines will understand.

Step 4 : Define the controller programme.

Knowing which instructions must be sent to which machines, the programme for the supervising controller can now be written. This is written in the supervising controller's language, and contains instructions on how and when to send the required instructions to the several machines, and probably further instructions to control timing, such as waiting for specific responses from machines.

QUESTION 6.

There were some good answers to this question. Most people clearly understood the general ideas, so got most of the marks. I thought it important that there should be some mention of interrupts, because most real-time systems do have something of the sort and the controller has to deal with it somehow. The notion that the controller schedules a set of identifiable tasks is also important.

A cyclic executive system is a scheduling method in which a list of tasks is executed regularly at a fixed frequency.

A master task is configured like this :

```
while true
do   begin
     await clock interrupt;
     task 1;
     task 2;
     task 3;
     .....
     end;
```

Interrupts other than the clock interrupt must be handled separately, and must not initiate any significant tasks; typically, they set flags which can be inspected by the appropriate task when it runs.

The advantages of the cyclic executive are :

1 : Simplicity. No special facilities are required, and it will run on a very simple system.

2 : Predictability. The processing load is well known, and provided that the sum of the execution times of the tasks is less than the time between the clock interrupts performance can be guaranteed.

3 : Reliability. There's nothing to go wrong, and nothing unexpected can happen.

The disadvantages of the cyclic executive are :

1 : Possible inefficiency : Every task is initiated at each cycle, and if some only require attention every ten cycles, say, the overhead can be significant.

2 : Slow response : If an interrupt requires significant attention as soon as it happens, the cyclic executive is not appropriate.

3 : Not good for bursty workloads : If the tasks are of unpredictable execution time, varying from cycle to cycle, it is necessary to cater for the most pessimistic combination of maximum times if reliable operation is to be guaranteed.

QUESTION 7.

1 : Variables of type time (absolute and interval) and corresponding arithmetic operations, and a way of reading the current time;

2 : A timed wait instruction – `wait for <time interval>` or `wait until <absolute time >`.

Variants of timed waits abounded; typed variables didn't. Ways of reading the time were mentioned.

For the ILIAD `when` instruction :

> The variable `counter` is a simple integer variable, so there is no reason why its changes in value should be associated with any external event. The compiler must either set up a clocked polling loop which inspects the value of `counter` "sufficiently frequently", or it must insert code to test the value every time that `counter` is changed in the programme.

For the PEARL `when` instruction :

> The identifier `alarm` is declared as an interrupt, so the `<instruction>` must be set up as an interrupt service procedure attached to the interrupt.

I looked for something equivalent to an interrupt service routine, but didn't find many. I don't think it's avoidable.

A few people assumed that because the variable was called "counter" they had to set up a counter. I didn't say that. In fact, both the Iliad and Pearl examples were taken directly from the programme examples we looked at in the course, and I just used the names which appeared there.

QUESTION 8.

An ERROR is a mistake made in designing or constructing a system.
A FAULT is the result of the error which remains in the system and which might cause misbehaviour. Faults can also be accidental.
A FAILURE is the undesired event, which might be caused by a fault.

Most people seemed to know that bit, though there was a certain shakiness around faults.

SHALLOW INTELLIGENCE characterises failure managers in which the performance results from a process not involving analysis depending on the structure of the failed system. A failure initiates a search of the rule base to find a match with the current plant conditions; if a match is found, the corresponding solution is retrieved.

DEEP INTELLIGENCE is found in failure managers which attempt to determine the cause of a failure by reasoning processes involving the causal relationships between events which occur in the plant, which can be listed or derived from a plant model.

I did look for something related to rule-based and model-based approaches in the definitions. The words didn't have to be there, but I wanted something indicating the differences between the methods.

Shallow intelligence failure managers will only handle failures which have been explicitly foreseen, or which have happened before and been recorded in the rule base. For such failures they are quite reliable, but for others they will fail, by producing either no diagnosis or a wrong diagnosis.
Deep intelligence failure managers will in principle handle any failure, but their performance is determined by the quality of the plant description which they use.

QUESTION 9.

This was an easy question, and very popular. Most people played safe, but there were one or two answers to the second part which suggested people were thinking about the issue.

1 : A robot is a general-purpose machine which can move relatively freely in space and has one or more "limbs" constructed as a series of articulated segments.

2 : A robot is a general-purpose machine which can move relatively freely in space and be programmed to carry out any of a wide range of tasks.

A directly operated arm-like manipulator conforms to the first definition, but not the second.

(Any sensible answer will do, but must contain notions of generality and free motion.)

Almost everybody knew that bit. Most followed the party line laid down in lectures and notes, but a couple had other, and interesting, ideas.

Any plausible answer to the final part will do, and will be assessed on its merits.

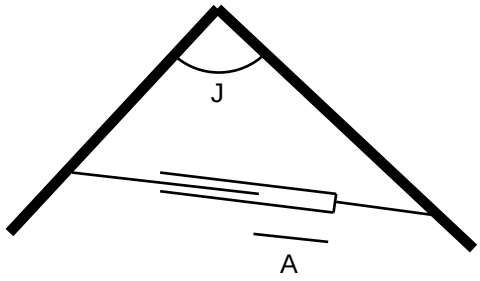
I looked for : a practicable answer (no positronic brains or human conversation – all were practicable); consistency (so if you said it was autonomous and controllable I wanted to know how these were fitted together – not quite all did that); generality and space (without these it's just a machine; lots of machines react to their environments,. for example).

QUESTION 10.

A robot arm is composed of several rigid members connected together by flexible couplings of various sorts. The rigid members are the links; the couplings are the joints.

A lot of floundering was evident in the answers. People who drew pictures won heavily.

A joint coordinate is a number which identifies the current position of a joint; an actuator cooordinate is a number which identifies the current state of an actuator. They might be different, because the connection between joint and actuator might not be direct. An example is the use of a hydraulic actuator to control a joint angle; here the two coordinates are likely to be the joint angle and the hydraulic element's linear displacement.



Often not well done. I don't know why.

The D-H coordinates of a link are measured from the mutual perpendicular to the axes of its two joints. These are the *link length*, which is the length of the perpendicular, and the *link twist*, which is the angle between the two axes. These two parameters are fixed by the nature of the link. When two links are connected, the axes of the connected joints are necessarily coincident, and this mutual axis is necessarily the common perpendicular to the two perpendiculars representing the links. The displacement between the two perpendiculars is the *joint displacement*, and the angle between them is the *joint angle*. One or both of these parameters is variable. When put together, the axes and perpendiculars for the full set of jointed links forms a chain of perpendicular lines, with a configuration fully determined by the values of the parameters.

Nobody really did this satisfactorily.

A joint with a fixed joint displacement and variable joint angle is called a *revolute joint* :

A joint with a variable joint displacement and fixed joint angle is called a *prismatic joint* :

This seemed to get missed out quite often. Usually done well if done at all.

QUESTION 11.

The behaviour of an autonomous robot is necessarily impossible to predefine (or it wouldn't be autonomous), so it might be required at any time to move in any direction. In contrast, the motions of a simple machine are fully predefined; it is known beforehand exactly what the motions will be, and under what circumstances they will occur. Sensors for a simple machine can therefore be placed precisely where they are needed, and linked more or less directly to the programme module which will deal with the part of the motion concerned. The autonomous robot might at any time require information about any part of its environment, so it is useful to collect sensory information about the world continuously and store it as a record of the robot's best interpretation of all it can sense at the moment. This record is the world view. Also, without such continuous monitoring, events might occur in the world without the robot's becoming aware of them.

I rather liked people to make the point that for a simple machine a few finite localised sensors would be sufficient.

When moving freely, a robot's motion is unconstrained, and can be in any accessible direction. In compliant motion, the robot must move in a way which coordinates its motion with the presence of

objects in the world around it. A simple example is a robot's gripper grasping a fixed object; unless it is very precisely placed before making the grasp, it is usually necessary to move the gripper a little to align it with the object grasped. To move appropriately, the robot must determine when it touches the object, and with what part of the gripper; some sort of contact or force sensor is needed to give this information. Given this sensory input, it can move to keep the contact pressure small as it closes the gripper further.

"Compliance" is usually used in the sense of conforming to constraints on motion, not in as a term for general sensible behaviour. Things like not squeezing eggs, carrying cups the right way up, and such come under the heading of "world knowledge", or some such term.

QUESTION 12.

The first mass-appeal question – all but one did it, and most got most of the marks. (The exceptions were mostly those who missed out the second part of the question.)

On-line programming, otherwise called teaching, is carried out using the robot itself as a tool. Typically, the robot is manipulated (by hand, or with the assistance of a teaching pendant or similar device) into a series of positions, which the robot system records by reading the internal sensors. The sequence can then be played back as and when required.

ADVANTAGES : Easy; accurate.
DISADVANTAGES : Expensive (you might have to stop a production line).

Off-line programming resembles conventional computer programming; instructions are written in some programming language, and executed by a compiler or interpreter when complete.

ADVANTAGES : Flexible – much easier to incorporate conditional behaviour.
DISADVANTAGES : Very difficult to be precise without using the robot.

Actuator level :	Expressed in terms of actuator coordinates; "machine language".
Joint level :	Expressed in terms of joint coordinates; "assembly language".
Manipulator level :	World cooordinates; simple movement instructions – MOVE, GRASP, etc.
Task level :	Instructions in terms natural to the task.
Object level :	Description of final product given; the system works out the detailed instructions.

QUESTION 13.

The control connections of the interface adapter chip are connected (usually) to the low-order bits of the computer's address bus, and the chip's data connections attached to the data bus. The high-order address bits are inspected by a comparison chip, which send a select signal to the adapter chip when the high order bits match a preset pattern.

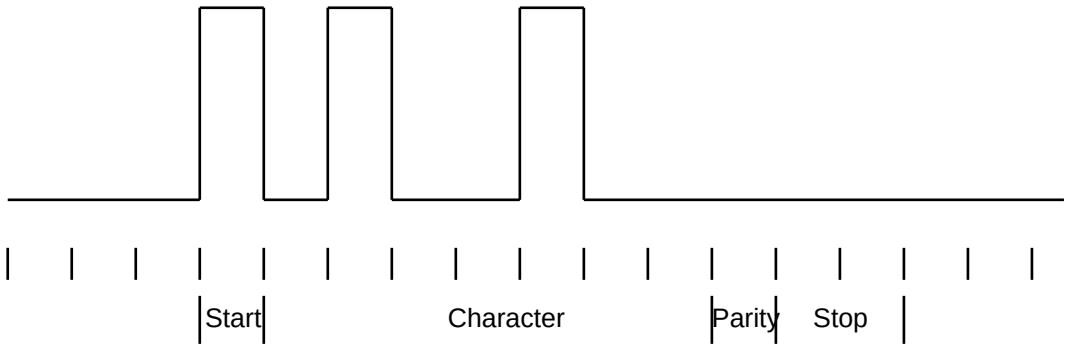
Not everyone remembered the select signal. I got lots of stuff about how you read and write, for which I didn't ask.

An interface adapter is useful because it can attend to low-level communication functions without interfering with the main computer. Interrupts are reduced, and some simple operations might be completely taken care of by the adapter.

Rather few people got that answer, but most gave something else which was acceptable. A few just ignored this bit.

QUESTION 14.

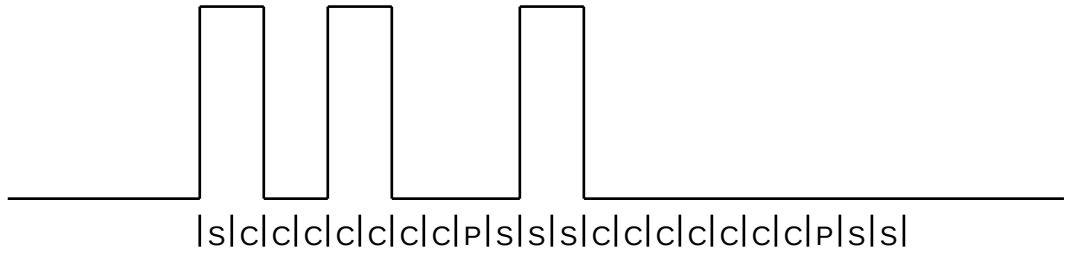
(a)



(b) 1101101

Low is 1; high is 0.

(c)

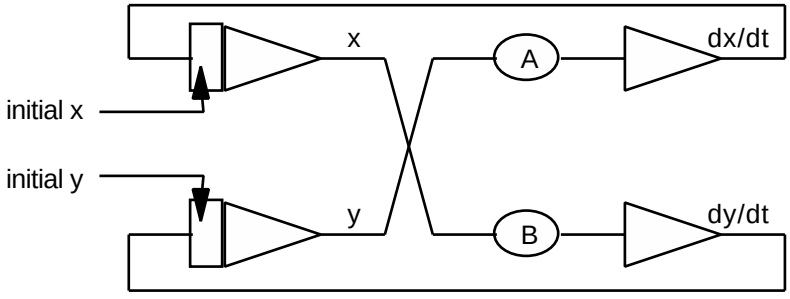


The first character is identified as 1100110, but the parity bit is wrong so that character is rejected. What happens next depends on just when the circuitry starts looking again. If it waits for the official end of the character, as shown in the diagram, it first gets a framing error because there is only one stop bit, but then receives character 1111111, and the parity is wrong again.

QUESTION 15.

An analogue computer is composed of adders, integrators, and potentiometers, with some means of connecting them together in arbitrary ways. It must also have means of setting initial values of integrator outputs at the beginning of an experiment.

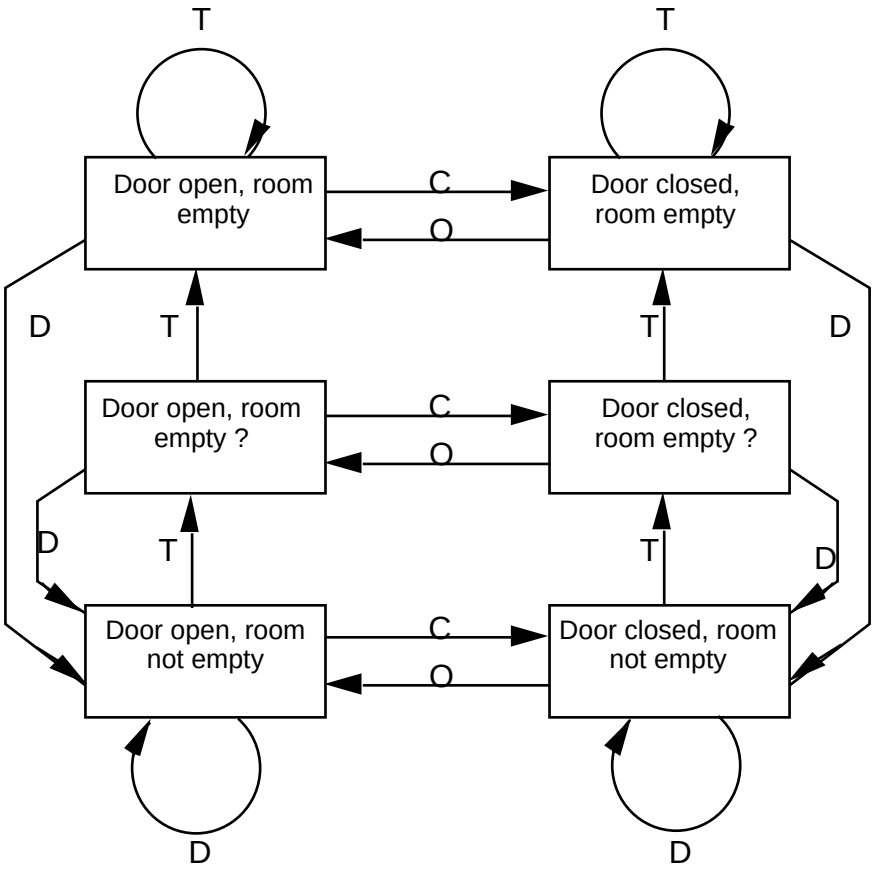
The output is inverted because the operation of the adder depends on the sum of output and input being very close to zero. It would be easy to add an inverter stage, but that would approximately double the number of components in the system and potentially introduce additional errors.



This was last year's question. I thought that was all right, because no one answered it last year, and I didn't want to waste it. Most people who attempted it got very good marks; a few were unsure about the inversion of sign.

QUESTION 16.

Clock signal :	T
Movement detector :	D
Door open :	O
Door closed :	C



Hm. Lots of attempts, not many very good answers. A significant defect was a strong wish to describe the states in terms of whether or not the light was on; that's a consequence of the state, not part of its definition. Apart from that, which was misleading because it led to unhelpful state descriptions, few people noticed that you could open and close a door but still have no evidence that there was anyone in the room, so counting the ticks is independent of the behaviour of the door. (That might be a faulty specification, but that's what the specification says.)

Note that a state diagram should show the transition from every state for every event.